

3.4)return listener 消息处理机制

一:Return Listener是用来处理一些不可路由的消息

1.1:我们的消息生产者, 通过把消息投递到exchange上, 然后通过routingkey 把消息路由到某一个队列上, 然后我们消费者通过队列消息侦听, 然后进行消息消费处理.

以上会出现的情况

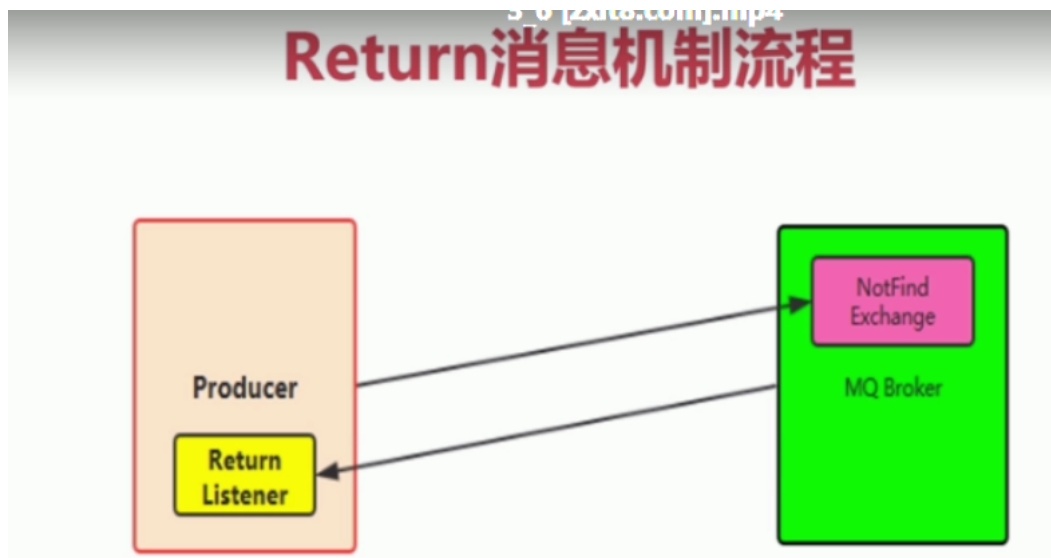
情况一: broker中根本没有对应的exchange交换机来接受该消息

情况二:消息能够投递到broker的交换机上, 但是交换机根据routingKey 路由不到某一个队列上.

针对上述二种情况 我们就需要return listener来处理这种不可达的消息.

处理一; 若在消息生产端 的mandatory设置为true 那么就会调用生产端ReturnListener 来处理,

处理二; 若消息生产端的mandatory设置为false(默认值也是false) 那么mq-broker就会自动删除消息



代码演示:(生产端)

```
public static void main(String[] args) throws IOException, TimeoutException {
    ConnectionFactory connectionFactory = new ConnectionFactory();
    connectionFactory.setVirtualHost("/");
    connectionFactory.setHost("47.104.128.12");
    connectionFactory.setPort(5672);
    Connection connection = connectionFactory.newConnection();
    Channel channel = connection.createChannel();
    //设置return listener
    channel.addReturnListener(new AngleReturnListener());
    //可达消息
    channel.basicPublish("test.return.exchange", "test.return.key", false, null, "return listener test".getBytes());
    //不可达消息 调用return listener
    channel.basicPublish("test.return.exchange", "test.return.key1", true, null, "return listener test2".getBytes());
    //不可达消息,mq-broker自动删除模式
    channel.basicPublish("test.return.exchange", "test.return.key2", false, null, "return listener test3".getBytes());
}
```

```
}
```

代码演示(消费端)

```
public static void main(String[] args) throws IOException, TimeoutException, InterruptedException {
    ConnectionFactory connectionFactory = new ConnectionFactory();
    connectionFactory.setVirtualHost("/");
    connectionFactory.setHost("47.104.128.12");
    connectionFactory.setPort(5672);
    Connection connection = connectionFactory.newConnection();
    Channel channel = connection.createChannel();
    channel.exchangeDeclare("test.return.exchange","direct",true,true,false,null);
    channel.queueDeclare("test.return.queue",true,false,true,null);
    channel.queueBind("test.return.queue","test.return.exchange","test.return.key");
    QueueingConsumer queueingConsumer = new QueueingConsumer(channel);
    channel.basicConsume("test.return.queue",true,queueingConsumer);
    while (true) {
        QueueingConsumer.Delivery delivery = queueingConsumer.nextDelivery();
        System.out.println(new String(delivery.getBody()));
    }
}
```

代码演示(return listener)

```
public class AngleReturnListener implements ReturnListener {
    @Override
    public void handleReturn(int replyCode, String replyText, String exchange, String routingKey, AMQP.BasicProperties prop) {
        System.out.println("记录不可达消息.....");
        System.out.println("replaycode="+replyCode);
        System.out.println("replyText="+replyText);
        System.out.println("exchange="+exchange);
        System.out.println("routingKey="+routingKey);
        System.out.println("properties="+properties);
        System.out.println("body="+new String(body));
    }
}
```